12-Hour Clock using Verilog

Self Project

Name: Sajid Iquebal Roll No: **22M1134**

DESIGN DESCRIPTION:

Create a set of counters suitable for use as a 12-hour clock (with am/pm indicator). Your counters are clocked by a fast-running clk, with a pulse on ena whenever your clock should increment (i.e., once per second).

TOP MODULE

```
module
   top_module(input clk,
   input reset,
   input ena,
   output pm,
   output [7:0] hh,
   output [7:0] mm,
   output [7:0] ss
);

   // Declare registers for each BCD digit
   reg [3:0] sec0, sec1;
   reg [3:0] min0, min1;
   reg [3:0] hour0, hour1;

   // AM/PM indicator logic
   always  @(posedge  clk)
   begin
      if     (reset)
         pm <= 0;
      else if ((hour0 == 4'd1) && (hour1 == 4'd1) && (min1 == 4'd5) && (min0 == 4'd9) &&
(sec1 == 4'd5) && (sec0 == 4'd9))
         pm <= ~pm;
   end

   // Seconds unit place counter
   always  @(posedge   clk)
   begin
      if (reset)
         sec0 <= 0;
      else if (ena)
      begin
         if (sec0 == 4'd9)
            sec0 <= 4'd0;
```

```
      else
         sec0 <= sec0 + 1;
   end
end

// Seconds tens place counter
always @(posedge clk)
begin
   if (reset)
      sec1 <= 0;
   else if (ena)
   begin
      if ((sec1 == 4'd5) && (sec0 == 4'd9))
         sec1 <= 4'd0;
      else if (sec0 == 4'd9)
         sec1 <= sec1 + 1;
   end
end

// Minutes unit place counter
always   @(posedge   clk)
begin
   if (reset)
      min0 <= 0;
   else if (ena)
   begin
      if ((min0 == 4'd9) && (sec1 == 4'd5) && (sec0 == 4'd9))
         min0 <= 4'b0;
      else if ((sec1 == 4'd5) && (sec0 == 4'd9))
         min0 <= min0 + 1;
   end
end

// Minutes tens place counter
always   @(posedge    clk)
begin
   if (reset)
      min1 <= 0;
   else if (ena)
   begin
      if ((min1 == 4'd5) && (min0 == 4'd9) && (sec1 == 4'd5) && (sec0 == 4'd9))
         min1 <= 4'd0;
      else if ((min0 == 4'd9) && (sec1 == 4'd5) && (sec0 == 4'd9))
         min1 <= min1 + 1;
   end
end

// Hours unit place counter
```

```
    always @(posedge clk)
    begin
        if (reset)
            hour0 <= 4'd2;
        else if (ena)
        begin
            if ((sec1 == 4'd5) && (sec0 == 4'd9) && (min1 == 4'd5) && (min0 == 4'd9))
            begin
                if ((hour0 == 4'd9) && (hour1 == 4'd0))
                    hour0 <= 4'd0;
                else if ((hour0 == 4'd2) && (hour1 == 4'd1))
                    hour0 <= 4'd1;
                else
                    hour0 <= hour0 + 1;
            end
        end
    end

    // Hours tens place counter
    always @(posedge clk)
    begin
        if (reset)
            hour1 <= 4'd1;
        else if (ena)
        begin
            if ((hour1 == 4'd1) && (hour0 == 4'd2) && (sec1 == 4'd5) && (sec0 == 4'd9) &&
(min1 == 4'd5) && (min0 == 4'd9))
                hour1 <= 4'd0;
            else if ((sec1 == 4'd5) && (sec0 == 4'd9) && (min1 == 4'd5) && (min0 == 4'd9) &&
(hour0 == 4'd9))
                hour1 <= hour1 + 1;
        end
    end

    //  Assign outputs
    assign ss = {sec1, sec0};
    assign mm = {min1, min0};
    assign hh = {hour1, hour0};

endmodule
```

TESTBENCH

```
module tb_top_module;

    // Inputs
    reg clk;
    reg reset;
    reg ena;

    // Outputs
    wire   pm;
    wire [7:0] hh;
    wire [7:0] mm;
    wire [7:0] ss;

    // Instantiate the clock module
    top_module dut (
        .clk(clk),
        .reset(reset),
        .ena(ena),
        .pm(pm),      // No need to connect pm in the testbench
        .hh(hh),
        .mm(mm),
        .ss(ss)
    );

   wire observed_pm; // Additional wire to observe pm
    assign observed_pm = pm;

    // Clock generation
    reg clk_period = 10; // Adjust this value to set the clock period
    always #((clk_period)/2) clk = ~clk;

    // Testbench stimulus
    initial begin
        reset = 1;
        ena = 0;

        // Reset and wait for a few clock cycles
        #10 reset = 0;

        // Enable the clock module
        ena = 1;
// Test various scenarios
        #100;
        // Display the time and AM/PM value
        $display("Time: %d:%d:%d %s", hh, mm, ss, (observed_pm) ? "PM" : "AM");
```

```
        // Wait for some time to see the clock changes
        #200000;

        // Disable the clock module
        ena = 0;

        // Finish simulation
        $finish;
    end

endmodule
```

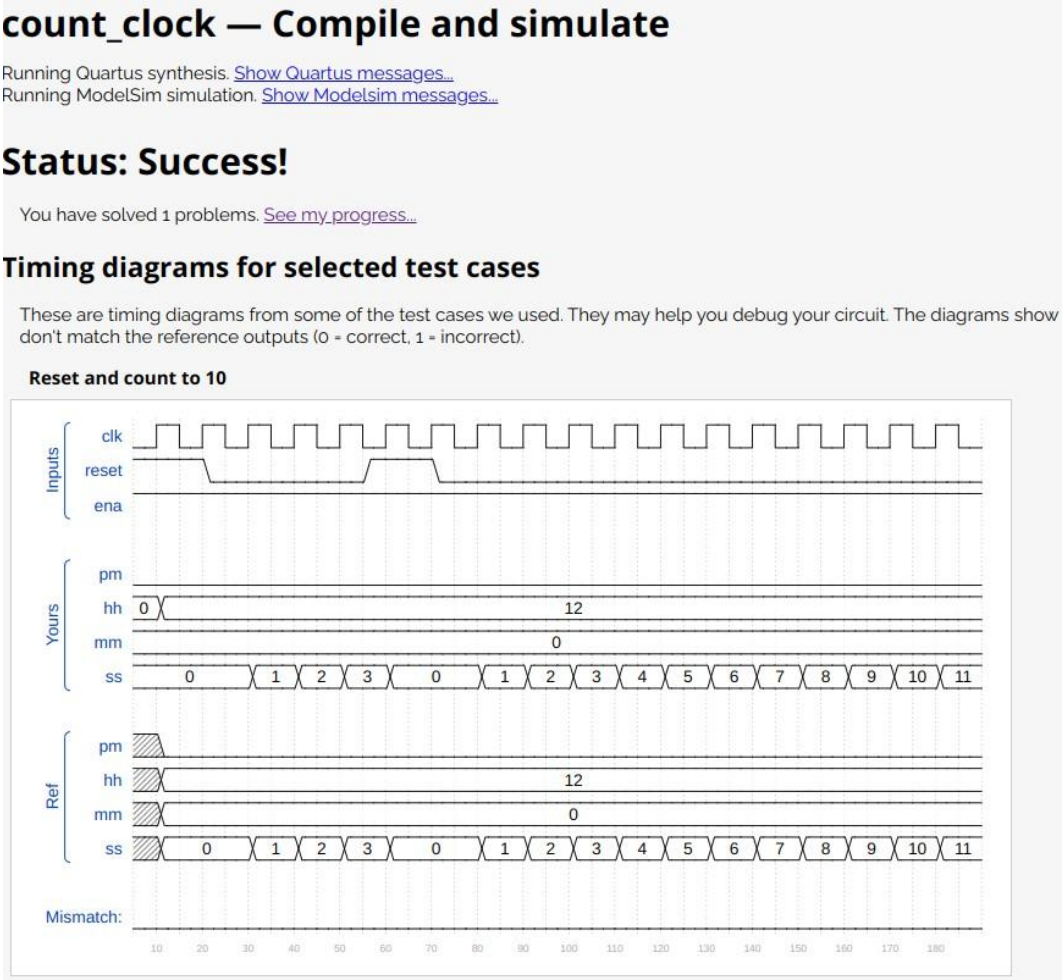

Fig : Verification of design on HDLBits platform using Quartus synthesis and Modelsim simulation.